Learning Image Matching by Simply Watching Video

Gucan Long[1,2(✉)], Laurent Kneip[2,3], Jose M. Alvarez[2,4], Hongdong Li[2,3], Xiaohu Zhang[1], and Qifeng Yu[1]

[1] National University of Defense Technology, Changsha, People's Republic of China
gucan.long@gmail.com
[2] Australian National University, Canberra, Australia
laurent.kneip@anu.edu.au
[3] Australian Centre of Excellence for Robotic Vision, Canberra, Australia
[4] Data61, CSIRO, Canberra, Australia

Abstract. This work presents an unsupervised learning based approach to the ubiquitous computer vision problem of image matching. We start from the insight that the problem of frame interpolation implicitly solves for inter-frame correspondences. This permits the application of analysis-by-synthesis: we first train and apply a Convolutional Neural Network for frame interpolation, then obtain correspondences by inverting the learned CNN. The key benefit behind this strategy is that the CNN for frame interpolation can be trained in an unsupervised manner by exploiting the temporal coherence that is naturally contained in real-world video sequences. The present model therefore learns image matching by simply "watching videos". Besides a promise to be more generally applicable, the presented approach achieves surprising performance comparable to traditional empirically designed methods.

Keywords: Image matching · Unsupervised learning · Analysis by synthesis · Temporal coherence · Convolutional neural network

1 Introduction

We are experiencing a tremendous success of deep learning in almost all research areas of computer vision. However, for most of the time, deep models are trained by relying on man-made supervising signals, which are all too often prepared through a tedious, expensive manual labeling process. Many researchers therefore believe that a more promising paradigm is given by unsupervised learning, as most of the readily available data simply comes in unlabeled form. This work contributes to this direction by providing an unsupervised solution to the ubiquitous vision problem of image matching. Specifically, relying on only natural video sequences, the present model is able to learn the ability of establishing 2D-2D correspondences across consecutive frames.

This work was conducted while G. Long was a visiting student at the ANU, supported by the China Scholarship Council (CSC), and supervised by L. Kneip.

B. Leibe et al. (Eds.): ECCV 2016, Part VI, LNCS 9910, pp. 434–450, 2016.
DOI: 10.1007/978-3-319-46466-4_26

Fig. 1. We train a deep convolutional network for frame interpolation, which can be done without manual supervision by exploiting the temporal coherence that is naturally contained in real-world video sequences. The learned CNN is then used to compute a sensitivity map for each output pixel. This sensitivity map, i.e. the gradients w.r.t. the input, indicates how much each input pixel influences a particular output pixel. The two input pixels (one per input frame) that have the maximum influence are considered as an image correspondence (i.e. a match). Though indirect, the resulting model learns how to perform dense correspondence matching by simply watching video.

Our key insight lies in the understanding that frame interpolation implicitly solves for dense correspondences between the input image pair. It is well known that dense matching can be regarded as a sub-problem of frame interpolation, as the interpolation could be immediately generated by correspondence-based image warping once dense inter-frame matches are available [3]. It then comes as no surprise that if we were able to train a deep neural network for frame interpolation, its application would implicitly also generate knowledge about dense image correspondences. Retrieving this knowledge is known as *analysis by synthesis* [42], a paradigm in which learning is described as the acquisition of a measurement synthesizing model, and inference of generating parameters as model inversion once correct synthesis is achieved. In our context, *synthesis* simply refers to frame interpolation. We then, for the *analysis* part, show that the correspondences can be recovered from the network through gradient back-propagation, which produces sensitivity maps for each interpolated pixel. The procedure is summarized in Fig. 1, explaining how the reciprocal mapping between frame interpolation and dense correspondences is encoded in the forward and backward propagation through one and the same network architecture. We call our approach MIND, which stands for Matching by INverting[1] a Deep neural network.

The key benefit of MIND lies in the fact that the deep convolutional network for frame interpolation can be trained from ordinary video sequences without any man-made ground truth signals. The training data in our case is given by triplets of images, each one consisting of two input images and one output image that represents the ground-truth interpolated frame. A correct example

[1] The term of *inverting* is read as *back-propagation* through the given deep neural network.

of a ground truth output image is an image that—when inserted in between the input pair of images—forms a *temporally coherent* sequence of frames. Such temporal coherence is naturally contained in regular video sequences, which allows us to simply use triplets of sequential images from almost arbitrary video streams for training our network. The first and the third frame of each triplet are used as inputs to the network, and the second frame as the ground truth interpolated frame. Most importantly, since the inversion of our network returns frame-to-frame correspondences, it therefore learns how to do image matching without any requirement for manually designed models or expensive ground truth correspondences. In other words, the presented approach learns image matching by simply "watching videos".

The paper is organized as follows. Section 2 reviews relevant prior work. Section 3 explains the present *analysis-by-synthesis* approach, including both the *analysis* part of how MIND works and the *synthesis* part of the deep convolutional architecture for frame interpolation. Section 4 demonstrates the surprising performance for the present purely unsupervised learning approach, which is comparable to several traditional empirically designed methods. Section 5 finally discusses our contribution and provides an outlook onto future work.

2 Related Work

Deep learning meets image matching: Image matching is a classical problem in computer vision. Here we limit the discussion to recent works that address image matching through learning based approaches. Roughly speaking, there exist two lines of research for this topic: the first one consists of making use of features or representations learned by deep neural networks, which are either originally trained for other tasks such as object recognition [13,26], or specially designed and trained for the purpose of image matching [1,21,33]. The second major line of research employs deep neural networks to compute the similarity between image patches [30,43,44]. In contrast to our work, the cited contributions mainly address sub-modules of image matching (feature extraction or matching cost computation), rather than providing end-to-end solutions. An exception is given by FlowNet [14], which presents an interesting deep learning based approach for dense optical flow computation. It does however depend on ground truth flow for training the network.

Temporal coherence learning: Unsupervised learning is a broad topic in the field of machine learning. Our discussion here focuses on works that exploit temporal coherence in natural videos, sometimes also called *temporal coherence learning* [4,29,41]. As a recent representative work, Wang et al. [39] exploit temporal coherence by visual tracking in videos, and report that the learned representation achieves competitive performance compared to some supervised alternatives. While temporal coherence learning mostly aims at learning features or representations, some recent works on reconstructing and predicting video frames in an unsupervised setting [31] are closely related to our work as well. Srivastava et al. [35] use an encoder LSTM to map input sequences into a fixed

length representation, and use the latter for reconstructing the input or even predicting future frames. Goroshin et al. [17] consider videos as one-dimensional, time-parametrized trajectories embedded in a low dimensional manifold. They train deep feature hierarchies that linearize the transformations observed in natural video sequences for the purpose of frame prediction. Though related to our work, these works are not aiming at image matching. It will be interesting to apply our concept of matching by inverting to the above models for temporal coherence learning.

Inversion of artificial neural network: Note that inverting a learned network is traditionally defined as reconstructing the input from the output of an artificial neural network [22]. Mahendran et al. [27] and Dosovitskiy et al. [10] apply this concept to understand what information is preserved by a network. In our context, *inverting a network* means *back-propogation through a learned network in order to obtain the gradient map with respect to the input signals.* Interestingly, the idea has already been introduced in the work of Simonyan et al. [34], emphasizing that the retrieved sensitivity maps may serve to identify image-specific class saliency. Similarly, Bach et al. [2] employ gradient maps as a measure for the contribution of single pixels to nonlinear classifiers, thus helping to explain how decisions are made.

3 Methodology

The *analysis by synthesis* approach for dense image matching is described in this section: we first explain the *analysis* part, i.e. how to obtain correspondences given the trained neural network and the interpolated image. For the *synthesis* part, we describe here the detailed architecture of the deep convolutional network designed for frame interpolation.

3.1 Matching by Inverting a Deep Neural Network

Assuming that we have a well trained deep neural network for frame interpolation in our hand, the core technical question behind our work is how to recover the correspondences between the input pair of images from there. As explained previously, dense correspondence matching may be regarded as a sub-problem of frame interpolation, which is why we should be able to trace back the matches starting from the interpolated frame generated during the forward-propagation through the trained network. Our task then consists of back-tracking each pixel in the output image to exactly one pixel in each of the two input images. Note that this back-tracking does not mean reconstructing input images from the output one. Instead, we only need to find the pixels in each input image which have the maximum influence to each pixel of the output image.

We perform back-tracking by applying a technique similar to the one adopted by Simonyan et al. [34]. For each pixel in the output image, we compute the gradient of its value with respect to each input pixel, thus telling us how much it

is under the influence of individual pixels at the input. The gradient is computed based on back-propagation, and leads to sensitivity or influence maps at the input of the network.

From a more formal perspective, our approach may be explained as follows. Let $\mathbf{I}_2 = \mathcal{F}(\mathbf{I}_1, \mathbf{I}_3)$ denote a non-linear function (i.e. the trained deep neural network) that describes the mapping from two input images $\mathbf{I}_1$ and $\mathbf{I}_3$ to an interpolated image $\mathbf{I}_2$ lying approximately at the "center" of the input frames. Thinking of $\mathcal{F}$ as a vectorial mapping, it can be split up into $h \times w$ non-linear sub-functions, each one producing the corresponding pixel in the output image

$$\mathcal{F}(\mathbf{I}_1, \mathbf{I}_3) = \begin{pmatrix} f^{11}(\mathbf{I}_1, \mathbf{I}_3) & \dots & f^{1w}(\mathbf{I}_1, \mathbf{I}_3) \\ \vdots & & \vdots \\ f^{h1}(\mathbf{I}_1, \mathbf{I}_3) & \dots & f^{hw}(\mathbf{I}_1, \mathbf{I}_3) \end{pmatrix}_{h \times w} . \tag{1}$$

In order to produce the sensitivity maps, we apply back-propagation to compute the Jacobian matrix with respect to each input image individually. The Jacobian with respect to the first image is given by

$$\frac{\partial \mathcal{F}(\mathbf{I}_1, \mathbf{I}_3)}{\partial \mathbf{I}_1} = \begin{pmatrix} \frac{\partial f^{11}(\mathbf{I}_1, \mathbf{I}_3)}{\partial \mathbf{I}_1} & \cdots & \frac{\partial f^{1w}(\mathbf{I}_1, \mathbf{I}_3)}{\partial \mathbf{I}_1} \\ \vdots & & \vdots \\ \frac{\partial f^{h1}(\mathbf{I}_1, \mathbf{I}_3)}{\partial \mathbf{I}_1} & \cdots & \frac{\partial f^{hw}(\mathbf{I}_1, \mathbf{I}_3)}{\partial \mathbf{I}_1} \end{pmatrix}_{h \times h \times w \times w} , \tag{2}$$

illustrating that this derivative results in one $h \times w$ matrix for each one of the $h \times w$ pixels at the output. The Jacobian with respect to $\mathbf{I}_3$ is given in a similar way. Let's define the absolute gradients of the output point (i, j) with respect to each one of the input images, and evaluated for the concrete inputs $\hat{\mathbf{I}}_1$ and $\hat{\mathbf{I}}_3$. They are given by

$$\begin{cases} \mathcal{G}^{i,j}_{\mathbf{I_1}}(\hat{\mathbf{I}}_1, \hat{\mathbf{I}}_3) = \text{abs} \left(\left. \frac{\partial f^{ij}(\mathbf{I}_1, \mathbf{I}_3)}{\partial \mathbf{I}_1} \right|_{\substack{\mathbf{I}_1 = \hat{\mathbf{I}}_1 \\ \mathbf{I}_3 = \hat{\mathbf{I}}_3}} \right) \\ \mathcal{G}^{i,j}_{\mathbf{I_3}}(\hat{\mathbf{I}}_1, \hat{\mathbf{I}}_3) = \text{abs} \left(\left. \frac{\partial f^{ij}(\mathbf{I}_1, \mathbf{I}_3)}{\partial \mathbf{I}_3} \right|_{\substack{\mathbf{I}_1 = \hat{\mathbf{I}}_1 \\ \mathbf{I}_3 = \hat{\mathbf{I}}_3}} \right) \end{cases} , \tag{3}$$

where abs replaces each entry of a matrix by its absolute value. The gradient maps produced in this way notably represent the seeked sensitivity or influence maps that may now serve in order to derive the coordinates of each correspondence. We notably extract the most responsible point in each gradient map, and connect those two points in order to return the correspondence.

In the spirit of unsupervised learning, we opted for the simplest possible choice, namely taking the coordinates of the maximum entry in $\mathcal{G}^{i,j}_{\mathbf{I_1}}(\hat{\mathbf{I}}_1, \hat{\mathbf{I}}_3)$ and $\mathcal{G}^{i,j}_{\mathbf{I_3}}(\hat{\mathbf{I}}_1, \hat{\mathbf{I}}_3)$, respectively. Let us denote these points with $c^{ij}_{\mathbf{I}_1}$ and $c^{ij}_{\mathbf{I}_3}$. By computing the two gradient maps for each point in the output image and extracting each time the most responsible point, we thus obtain the following two lists of points

$$\begin{cases} \mathcal{C}_{\mathbf{I}_1} = \left\{ c_{\mathbf{I}_1}^{ij} \right\} \\ \mathcal{C}_{\mathbf{I}_3} = \left\{ c_{\mathbf{I}_3}^{ij} \right\} \end{cases}, i = 1, \ldots, h, j = 1, \ldots, w \tag{4}$$

The set of correspondences $\mathcal{S}$ is then given by combining same-index elements from $\mathcal{C}_{\mathbf{I}_1}$ and $\mathcal{C}_{\mathbf{I}_3}$, eventually resulting in

$$\begin{aligned} \mathcal{S} &= \left\{ s^{ij} \right\}, i = 1, \ldots, h, j = 1, \ldots, w \\ &= \left\{ \left\{ c_{\mathbf{I}_1}^{11}, c_{\mathbf{I}_3}^{11} \right\}, \ldots, \left\{ c_{\mathbf{I}_1}^{hw}, c_{\mathbf{I}_3}^{hw} \right\} \right\}. \end{aligned} \tag{5}$$

3.2 Deep Neural Network for Frame Interpolation

The architecture of our frame-interpolation network is inspired by *FlowNet-Simple* as presented in Fischer et al. [14]. As illustrated in Fig. 2, it consists of a Convolutional Part and a Deconvolutional Part. The two parts serve as "encoder" and "decoder" respectively, similar to the auto-encoder architecture presented by Hinton and Salakhutdinov [20]. The basic block within the Convolutional Part—denoted Convolution Block—follows the common pattern of the convolutional neural network architecture:

INPUT –>[CONV –>PRELU] * 3 –>POOL –>OUTPUT.

The Parametric Rectified Linear Unit [19] is adopted in our work. Following the suggestions from VGG-Net [9], we set the size of the receptive field of all convolution filters to three—along with a stride and a padding of one—and duplicate **[CONV –>PRELU]** three times to better model the non-linearity.

The Deconvolution Part consists of Deconvolution Blocks, each one including a convolution transpose layer [38] and two convolution layers. The first one has a receptive field of four, a stride of two, and a padding of one. The pattern of the Deconvolution Block follows:

INPUT –>[CONVT –>PRELU] –>[CONV –>PRELU] * 2 –>OUTPUT.

In order to maintain fine-grained image details in the interpolation frame, we make a copy of the output features produced by Convolution Blocks 2, 3, and 4, and concat them as an additional input to the Deconvolution Blocks 4, 3, and 2, respectively. This concept is illustrated by the side arrows in Fig. 2, and similar ideas have already been used in prior work [11,14]. Recent works [18,36] indicate that the 'side arrows' may also help to better train the deep network.

It is easy to notice that our network is a fully convolutional one, thus allowing us to feed it with images of different resolutions. This is an important advantage, as different data-sets may use different height-to-width ratios. The output blob size for each block in our network is listed in Table 1.

4 Experiments

In this section, we first explain the implementation details behind MIND such as training data and loss function. The examples as proofs of concept for MIND

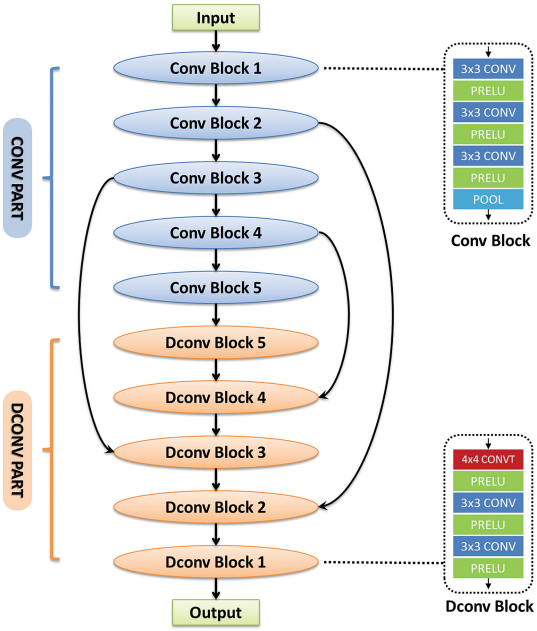

Fig. 2. Architecture of our network. The network takes 2 RGB images as an input to produce the interpolated RGB image. Please note that Dconv Block 4 takes the outputs from both Conv Block 2 and Dconv Block 5 as input. Dconv Block 3 and Dconv Block 2 have a similar input configuration. (Color figure online)

are introduced before a discussion on the generalization ability of the trained CNN. We finally evaluate MIND in terms of quantitative matching performance and compare it to traditional image matching methods.

4.1 Implementation Details

Training data: Quantity and quality of training data are crucial for training a deep neural network. However, our case is particularly easy as we can simply use huge amounts of real-world videos. In this work, we focus on training with the KITTI RAW videos [15] and Sintel videos[2] and show that the resulting learned network performs reasonably well. The network is first trained with the KITTI RAW video sequences which are captured by driving around the city of Karlsruhe, through rural areas and over highways. The dataset contains 56 image sequences with in total 16,951 frames. For each sequence, we take every three consecutive frames (both in forward and backward direction) as a training triplet, where the first and the third image serve as inputs to the network and the second image as the corresponding output. These images are then augmented by vertical flipping, horizontal flipping and a combination of both.

[2] Sintel, the Durian Open Movie Project. https://durian.blender.org/.

Table 1. The table lists the output blob size of each block in our network. Note that we stack two RGB images into one input blob, and thus the depth is 6. The output of the network is an RGB image and thus the depth equals to 3. The indicated widths are for the network trained on KITTI. The ones for the Sintel data are easily obtained, the only difference being that the input images are scaled to 256×128 rather than 384×128.

	Input	Conv1	Conv2	Conv3	Conv4	Conv5	Dconv5	Dconv4	Dconv3	Dconv2	Dconv1	Output
Depth	6	96	96	128	128	128	128	128	128	96	96	3
Height	128	64	32	16	8	4	8	16	32	64	128	128
Width	384	192	96	48	24	12	24	48	96	192	384	384

The total number of sample triplets is 133,921. We then fine-tune the network on examples selected from the original Sintel movie. We manually collected 63 video clips with in total 5,670 frames from the movie. After grouping and data augmentation we finally obtain 44,352 sample triplets. Note that, compared to the KITTI sequences which are recorded with relatively uniform velocity, the Sintel sequences represent more difficult training examples in the context of our work, as they contain a lot of fast and artificially rendered motion captured with a frame rate of only 24 fps. A significant portion of the Sintel samples therefore does not contain the required temporal coherence. We will discuss this issue further in Sect. 4.2.

Loss function: Several previous works [17,39] mention that minimizing the L2 loss between the output frame and the training example may lead to unrealistic and blurry predictions. We have not been able to confirm this throughout our experiments, but found that the Charbonnier loss $\rho(x) = \sqrt{(x^2 + \epsilon^2)}$ commonly employed for robust optical flow computation [37] leads to an improvement over the L2 loss. We employ it to train our network, with ϵ set to 0.1.

Training details: The training is performed using Caffe [23] on a machine with two K40c GPUs. The weights of the network are initialized by Xavier's approach [16] and optimized by the Adam solver [24] with a fixed momentum of 0.9. The initial learning rate is set to 1e-3 and then manually tuned down once ceasing of loss reduction sets in. For training on the KITTI RAW data, the images are scaled to 384×128. For training on the Sintel dataset, the images are scaled to 256×128. The batch size is 16. We run the training on KITTI RAW from scratch for about 20 epochs, and then fine-tuned it on the Sintel movie images for 15 epochs. We did not observe over-fitting during training, and terminated the training after 5 days.

Execution time: MIND can be applied to different scenarios (e.g. sparse or dense matching). We focus here on semi-dense image matching in order to obtain a result comparable with other methods. We compute the correspondences across the input images for each corner of a predefined raster grid of 4 pixels width in the interpolated image. Note that MIND currently depends on a large amount of computational resources as it performs back-propagation through the entire

network for every pixel that needs to be matched. For an image of size 384×128, each forward pass through our network takes 40 ms on a PC with K40c GPU, and each backward pass takes 158 ms. For each image pair, we need to perform one forward pass to first obtain the interpolation. We then need to perform $384 \times 128/4/4 = 3072$ backward passes to find the correspondences, resulting in a total of about 486 s (∼8 min).

4.2 Qualitative Examples for Interpolation and Matching

We demonstrate here the visual examples as proofs of concept for how the present approach works on both tasks of frame interpolation and image matching.

Fig. 3. Examples of frame interpolation (best viewed in colour). From left to right: example on KITTI, Sintel, ETH Multi-Person Tracking dataset [12] and Bonn Benchmark on Tracking [25], respectively. In each column, the first image is an overlay of the two input frames. The second one is the interpolated image obtained by our network. For the first example, we use the network trained on KITTI itself. For all others, we use the network fine-tuned on Sintel data. (Color figure online)

Examples of frame interpolation: We show the examples of frame interpolation in Fig. 3. The first two columns show the examples on KITTI and Sintel images which are taken from the validation data-sets originally collected for the purpose of monitoring the network training process. It can be seen that the trained CNNs cover the motion correctly for both KITTI and Sintel image pairs. It can furthermore be noticed that some fine-grained details are not preserved well in both examples, despite the special considerations in the architecture of the convolutional network (c.f. Sect. 3.2). Nevertheless, we would like to emphasize that the goal of the present work is not to provide a state-of-the-art frame-interpolation algorithm. As we will see, the preservation of fine-grained image details is in fact not necessarily an indicator for better quality image matching.

And for the goal of image matching, we will see that the preservation of perfect image details is in fact not necessary.

Examples for image matching: Here we present examples to demonstrate how MIND obtains correspondences given the trained CNNs for frame interpolation. The examples taken from KITTI and Sintel videos are shown in Fig. 4. By computing the gradient of manually marked pixels in the interpolated image, MIND successfully obtains correct correspondences between the 2 input images.

It can be seen that the correct correspondences are obtained even in some fast motion areas where fine-grained image details are missed, e.g. the area of the character's shaking hand in the Sintel example.

We further show one failure example taken from Sintel images. In Fig. 5, it can be observed that the interpolation fails as the motion of the small dragon and the character's hand have not been recovered correctly. It then comes as no surprise that MIND fails to extract correct matches for almost all of the selected points. However, it is worth to note that the No.4 match has better quality than others for which the corresponding gradient maps are less distinctive. The matching score/confidence returned by MIND is inspired by this behavior and defined as the ratio between the maximum gradient intensity and the mean gradient intensity within a small area around the maximal gradient location.

As illustrated in Sect. 4.4, the general performance of MIND, especially on KITTI images, is good. The failure example in Fig. 5 indicates an extreme case in the Sintel sequences dominated by fast and highly non-rigid motion in the scene.

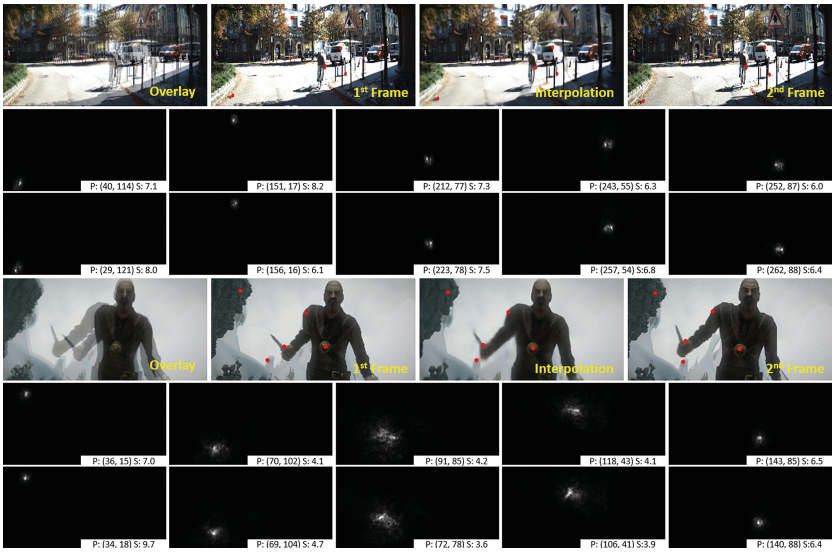

Fig. 4. Two matching examples for image pairs taken from the KITTI RAW video and the Sintel movie clip (best viewed in colour). For each example, the corresponding row of images shows input image 1, the interpolated image, and then input image 2 (from left to right). The red points mark five sample correspondences. The two rows below each example show the gradient/saliency maps for each match (from left to right) in each input image (maps for input image 1 on top, and maps for input image 2 in the bottom). The figures also indicate the coordinates of the maximal gradient location (P) along with the corresponding matching score (S). The matching score is defined as the ratio between the maximum gradient intensity and the mean gradient intensity within a 20×20 area around P. (Color figure online)

4.3 Generalization Ability of the Trained CNN

We first demonstrate the generalization ability of the trained CNN by applying it to images taken from the ETH Multi-Person Tracking dataset [12] and the Bonn Benchmark on Tracking [25], which have not been used for either training or fine-tuning. The results are shown in Fig. 3, from which we can see that the trained CNN again covers the motion correctly. It provides evidence that, by "watching videos", the present CNN is indeed learning the ability to interpolate frames and match images, rather than only "remember" the KITTI or Sintel-like images.

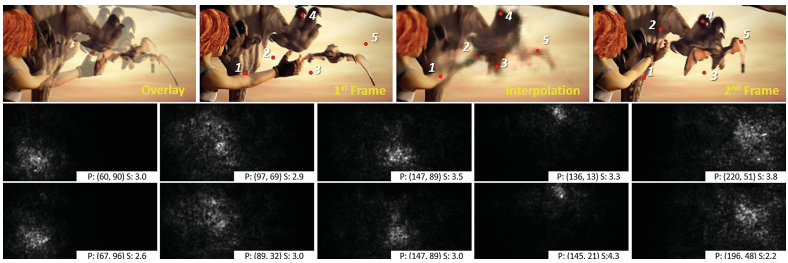

Fig. 5. Failure example of MIND for image pair taken from the Sintel movie clip (best viewed in colour). The gradient/saliency maps (from left to right) are for matches labelled as 1, 2, . . . , 5, respectively. (Color figure online)

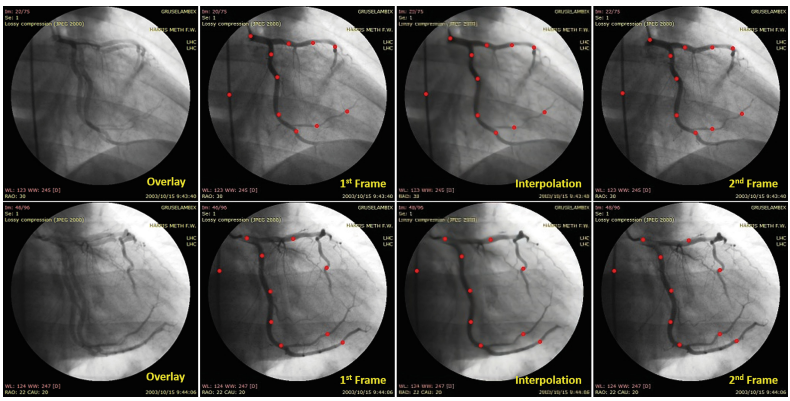

Fig. 6. Examples of MIND on DICOM images. There are two examples shown in different rows. For each example, the columns from left to right show the overlay of the input image-pair, the 1st input image, the interpolation returned by the CNN, and the 2nd input image, respectively. The red points in columns 2, 3 and 4 indicate the matches obtained by MIND. (Color figure online)

The generalization ability is further illustrated by applying MIND to DICOM images of coronary angiogram[3]. As a numerical evaluation of the generalization ability, we compare the CNN based interpolation results to traditional warp based interpolation method [3] using state-of-the-art optical flow, i.e. DeepFlow [40] and a recently proposed phase-based interpolation method [28]. The comparison is similar to the "Ground truth comparisons" outlined in [28]. The averaged SSD (sum of squared distances) for each method is 6.00, 6.23 and 5.55 respectively, suggesting that the trained CNN performs frame interpolation quantitatively well. Two examples are shown in Fig. 6. It can be seen that these images are substantially different from natural ones. Though failing to preserve perfect image details, the CNN, which is trained on natural images, performs impressively well on the DICOM images. The nice generalization ability of the CNN is underlined by results on both frame interpolation and image matching.

4.4 Quantitative Performance of Image Matching

We compare the matches produced by MIND against those of several empirically designed methods: the classical Kanade–Lucas–Tomasi feature tracker [5], HoG descriptor matching [7] (which is widely employed to boost dense optical flow computation), and the more recent DeepMatching approach [40] which relies on a multilayer convolutional architecture and achieves state-of-the-art performance. As observed in [40], comparing different matching algorithms is delicate because they usually produce different numbers of matches for different parts of the image. For the sake of a fair comparison, we adjust the parameters of each algorithm to make them produce as many as possible matches with an as homogeneous as possible distribution across the input images. For DeepMatching, we use the default parameters. For MIND, we extract correspondences for each corner of a uniform grid of 4 pixels width. For KLT, we set the minEigThreshold to 1e-9 to generate as many matches as possible. For HoG, we again set the pixel sampling grid width to 4. We then sort the matches according to suitable metrics[4] and select the same amount of "best" matches for each algorithm. In this way, the 4 algorithms produce the same numbers of matches with similar coverage over each input image.

The comparisons are performed on both KITTI [15] and MPI-Sintel [8] training sets where ground truth correspondences can be extracted from the available ground truth flow fields. We perform all of our experiments on the same image resolution than the one used by our network. On KITTI, the images are scaled to 384×128, and for MPI-Sintel, 256×128. We use the network trained on the

[3] The images are taken from a DICOM sample image set: http://www.osirix-viewer.com/datasets/. Alias Name: GRUSELAMBIX.

[4] For DeepMatching, we sort the matches according to the matching score given by the open source code [40]. For KLT, the metric is the error returned by the OpenCV implementation [6]. For HoG, we use the matching score defined in [7]. For MIND, the matching score is defined as the ratio between the maximum gradient intensity and the mean gradient intensity within a 20×20 area around the maximal gradient location.

Table 2. Matching performance on the KITTI 2012 flow training set. DeepM denotes DeepMatching. Metrics: Average Point Error (APE) (the lower the better), and Accuracy@T (the higher the better). Bold numbers indicate best performance, underlined numbers 2nd best.

	MIND	DeepM	HoG	KLT
APE	<u>4.695</u>	**3.442**	9.680	8.157
Accuracy@5	<u>0.716</u>	**0.835**	0.455	0.702
Accuracy@10	<u>0.915</u>	**0.953**	0.805	0.826
Accuracy@20	<u>0.981</u>	**0.987**	0.929	0.903
Accuracy@30	**0.993**	**0.993**	0.959	0.938

Table 3. Matching performance on the MPI-Sintel training set (Final pass). DeepM denotes DeepMatching. Metrics: Average Point Error (APE) (the lower the better), and Accuracy@T (the higher the better). Bold numbers indicate best performance, underlined numbers 2nd best.

	MIND	DeepM	HoG	KLT
APE	<u>5.838</u>	**3.240**	7.856	8.836
Accuracy@5	0.719	**0.875**	0.688	<u>0.808</u>
Accuracy@10	<u>0.876</u>	**0.951**	0.875	0.864
Accuracy@20	<u>0.948</u>	**0.977**	0.947	0.906
Accuracy@30	<u>0.967</u>	**0.986**	0.964	0.927

KITTI RAW sequences for the matching experiment on the KITTI Flow 2012 training set. We then use the network fine-tuned on Sintel movie clips for the experiments on the MPI-Sintel Flow training set. The 4 algorithms are evaluated in terms of the Average Point Error (APE) and the Accuracy@T. The latter is defined as the proportion of "correct" matches from the first image with respect to the total number of matches [32]. A match is considered correct if its pixel match in the second image is closer than T pixels to ground-truth.

As can be observed in Tables 2 and 3, DeepMatching produces matches with the highest quality in terms of all metrics and on both MPI-Sintel and KITTI sets. Notably, MIND performs very close to DeepMatching on KITTI and outperforms KLT tracking and HoG matching by a considerable amount in terms of Accuracy@10 and Accuracy@20. It is surprising to see that MIND—an unsupervised learning based approach—works so well. The performance on MPI-Sintel however drops a bit due to the difficulty of the contained artificial motion. Though the APE measure indicates better performance than HoG and KLT, it is only safe to conclude that MIND remains competitive in terms of overall performance on MPI-Sintel, which can be seen further in the next section.

4.5 Ability to Initialize Optical Flow Computation

To further understand the matching quality produced by MIND, we replace the DeepMatching part of DeepFlow [40] with MIND to see whether MIND matches are able to boost optical flow performance in a similar way than DeepMatching and HoG or KLT matches. Similar to the evaluation in [40], we feed DeepFlow with matches obtained by each matching method in the previous section. The parameters (e.g. the matching weight) of DeepFlow are tuned accordingly to make best use of the pre-obtained matches. Note that we scale down the input images to 384×128 for KITTI and 256×128 for MPI-Sintel. We then up-size the obtained flow field to the original resolution by bi-linear interpolation, to the end of comparing results in full resolution.

Table 4. Flow performance on KITTI 2012 flow training set (non-occluded areas). out-x refers to the percentage of pixels where flow estimation has an error above x pixels.

	MIND	DeepM	HoG	KLT	No match
APE	2.89	**2.63**	3.06	3.40	3.55
out-2	17.70 %	**17.09 %**	17.89 %	18.34 %	18.49 %
out-5	9.86 %	**9.18 %**	10.05 %	10.58 %	10.77 %
out-10	6.45 %	**5.84 %**	6.66 %	7.20 %	7.40 %

Table 5. Flow performance on MPI-Sintel flow training set. s0-10 is the APE for pixels with motions between 0 and 10 pixels. Similarly for s10-40 and s40+.

	MIND	DeepM	HoG	KLT	No match
APE	5.78	**4.80**	5.46	5.42	6.63
s0-10	**2.25**	2.84	3.65	3.22	2.47
s10-40	6.26	**6.08**	6.52	6.48	6.18
s40+	19.03	18.79	**17.38**	17.44	23.16

The results on the KITTI Flow 2012 training set are indicated in Table 4. It can be seen that using the matches obtained by any of the 4 algorithms improves the flow performance compared to the case where we use no matches for initialization. Notably, MIND again reaches closest performance to DeepMatching in terms of all metrics, thus underlining the good matching quality obtained by MIND (better than KLT and HoG and comparable to DeepMatching). Table 5 shows the results obtained on the MPI-Sintel training dataset. As in KITTI, the pre-obtained matches indeed help to improve the optical flow results especially in terms of the APE and s40+ metrics, while flow initialized by DeepMatching remains best overall. The results initialized from MIND matches however rank behind those initialized by HoG or KLT matches, which again suggests the importance of temporal coherence for training our network. The reason why KLT works better than in the evaluation presented in [40] is because we run KLT in the downscaled images rather than the full resolution ones, and this helps KLT to better deal with large displacements.

From the quantitative evaluations of matching and flow performance, it should be concluded that MIND works well on the KITTI Flow training set and achieves comparable performance to the state-of-the-art defined by DeepMatching. In the MPI-Sintel Flow training set, MIND still obtains comparable performance to the traditional HoG and KLT methods. The latter should still be interpreted as a good result especially considering that the quality of training data for the artificial and perhaps unrealistic Sintel images is insufficient. A closer look into the training data collected from Sintel video suggests that the assumption of temporal coherence does not hold well.

5 Conclusions

We have shown that the present work enables artificial neural networks to learn accurate image matching from only ordinary videos. Though MIND currently does not provide the required computational efficiency for applications in real-world scenarios, it promises a great potential for more natural solutions to further related problems. It is also our hope that the present work helps to promote the concept of *analysis by synthesis* towards a broad acceptance. Our future work

focuses on making the present approach more applicable in real-world scenarios, in terms of both computational efficiency and reliability.

Acknowledgment. L. Kneip's research is funded by ARC DECRA grant DE150-101365. The research of L. Kneip and H. Li is also funded by the ARC Centre of Excellence for Robotic Vision CE140100016. All authors gratefully acknowledge the support of the NVIDIA corporation for the donation of Tesla K40 GPUs. G. Long would like to give special thanks to Yuchao Dai, Stephen Gould and Anoop Cherian for the valuable discussions and feedback.

References

1. Agrawal, P., Carreira, J., Malik, J.: Learning to see by moving. arXiv preprint arXiv:1505.01596 (2015)
2. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS One **10**(7), e0130140 (2015)
3. Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M.J., Szeliski, R.: A database and evaluation methodology for optical flow. Int. J. Comput. Vis. **92**(1), 1–31 (2011)
4. Becker, S.: Learning temporally persistent hierarchical representations. In: Advances in Neural Information Processing Systems, pp. 824–830 (1997)
5. Bouguet, J.Y.: Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. Intel Corporation **5**(1-10), 4 (2001)
6. Bradski, G., Kaehler, A.: Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media Inc., Sebastopol (2008)
7. Brox, T., Malik, J.: Large displacement optical flow: descriptor matching in variational motion estimation. IEEE Trans. Pattern Anal. Mach. Intell. **33**(3), 500–513 (2011)
8. Butler, D.J., Wulff, J., Stanley, G.B., Black, M.J.: A naturalistic open source movie for optical flow evaluation. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7577, pp. 611–625. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33783-3_44
9. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: delving deep into convolutional nets. arXiv preprint arXiv:1405.3531 (2014)
10. Dosovitskiy, A., Brox, T.: Inverting convolutional networks with convolutional networks. arXiv preprint arXiv:1506.02753 (2015)
11. Eigen, D., Puhrsch, C., Fergus, R.: Depth map prediction from a single image using a multi-scale deep network. In: Advances in Neural Information Processing Systems, pp. 2366–2374 (2014)
12. Ess, A., Leibe, B., Schindler, K., Van Gool, L.: Robust multiperson tracking from a mobile platform. IEEE Trans. Pattern Anal. Mach. Intell. **31**(10), 1831–1846 (2009)
13. Fischer, P., Dosovitskiy, A., Brox, T.: Descriptor matching with convolutional neural networks: a comparison to sift. arXiv preprint arXiv:1405.5769 (2014)
14. Fischer, P., Dosovitskiy, A., Ilg, E., Häusser, P., Hazırbaş, C., Golkov, V., van der Smagt, P., Cremers, D., Brox, T.: Flownet: learning optical flow with convolutional networks. arXiv preprint arXiv:1504.06852 (2015)

15. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: the KITTI dataset. Int. J. Robot. Res. (IJRR) **32**, 1229–1235 (2013)
16. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: International Conference on Artificial Intelligence and Statistics, pp. 249–256 (2010)
17. Goroshin, R., Mathieu, M., LeCun, Y.: Learning to linearize under uncertainty. arXiv preprint arXiv:1506.03011 (2015)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385 (2015)
19. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. arXiv preprint arXiv:1502.01852 (2015)
20. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
21. Huang, G., Mattar, M., Lee, H., Learned-Miller, E.G.: Learning to align from scratch. In: Advances in Neural Information Processing Systems, pp. 764–772 (2012)
22. Jensen, C., Reed, R.D., Marks, R.J., El-Sharkawi, M., Jung, J.B., Miyamoto, R.T., Anderson, G.M., Eggen, C.J., et al.: Inversion of feedforward neural networks: algorithms and applications. Proc. IEEE **87**(9), 1536–1549 (1999)
23. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)
24. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
25. Klein, D.A., Schulz, D., Frintrop, S., Cremers, A.B.: Adaptive real-time video-tracking for arbitrary objects. In: IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 772–777, October 2010
26. Long, J.L., Zhang, N., Darrell, T.: Do convnets learn correspondence? In: Advances in Neural Information Processing Systems, pp. 1601–1609 (2014)
27. Mahendran, A., Vedaldi, A.: Understanding deep image representations by inverting them. arXiv preprint arXiv:1412.0035 (2014)
28. Meyer, S., Wang, O., Zimmer, H., Grosse, M., Sorkine-Hornung, A.: Phase-based frame interpolation for video. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1410–1418 (2015)
29. Mobahi, H., Collobert, R., Weston, J.: Deep learning from temporal coherence in video. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 737–744. ACM, New York (2009)
30. Park, M.G., Yoon, K.J.: Leveraging stereo matching with learning-based confidence measures. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 101–109 (2015)
31. Ranzato, M., Szlam, A., Bruna, J., Mathieu, M., Collobert, R., Chopra, S.: Video (language) modeling: a baseline for generative models of natural videos. arXiv preprint arXiv:1412.6604 (2014)
32. Revaud, J., Weinzaepfel, P., Harchaoui, Z., Schmid, C.: Deep convolutional matching. arXiv preprint arXiv:1506.07656 (2015)
33. Simo-Serra, E., Trulls, E., Ferraz, L., Kokkinos, I., Fua, P., Moreno-Noguer, F.: Discriminative learning of deep convolutional feature point descriptors. In: Proceedings of the International Conference on Computer Vision (ICCV) (2015)

34. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)
35. Srivastava, N., Mansimov, E., Salakhutdinov, R.: Unsupervised learning of video representations using LSTMs. arXiv preprint arXiv:1502.04681 (2015)
36. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks. arXiv preprint arXiv:1505.00387 (2015)
37. Sun, D., Roth, S., Black, M.J.: A quantitative analysis of current practices in optical flow estimation and the principles behind them. Int. J. Comput. Vis. **106**(2), 115–137 (2014)
38. Vedaldi, A., Lenc, K.: Matconvnet-convolutional neural networks for MATLAB. arXiv preprint arXiv:1412.4564 (2014)
39. Wang, X., Gupta, A.: Unsupervised learning of visual representations using videos. arXiv preprint arXiv:1505.00687 (2015)
40. Weinzaepfel, P., Revaud, J., Harchaoui, Z., Schmid, C.: Deepflow: large displacement optical flow with deep matching. In: 2013 IEEE International Conference on Computer Vision (ICCV), pp. 1385–1392. IEEE (2013)
41. Wiskott, L., Sejnowski, T.J.: Slow feature analysis: unsupervised learning of invariances. Neural Comput. **14**(4), 715–770 (2002)
42. Yildirim, I., Kulkarni, T., Freiwald, W., Tenenbaum, J.B.: Efficient and robust analysis-by-synthesis in vision: a computational framework, behavioral tests, and modeling neuronal representations. In: Annual Conference of the Cognitive Science Society (2015)
43. Zagoruyko, S., Komodakis, N.: Learning to compare image patches via convolutional neural networks. CoRR abs/1504.03641 (2015)
44. Žbontar, J., LeCun, Y.: Computing the stereo matching cost with a convolutional neural network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1592–1599 (2015)